bcopy()

Be careful with buffer size and termination

Sean Barnum, Cigital, Inc. [vita[1]]

2005-10-03

##### "Original Cigital Coding Rule in XML"

Mime-type: text/xml, ######: 5246 bytes

Rule Accuracy

False Positives

Attack Categories

- Malicious Input
- Denial of Service

Vulnerability Categories

- Buffer Overflow
- No Null Termination

Software Context

String Management

Description

bcopy (const void *src, void *dest, int n) - copies the first n bytes of the source string src to the destination string dest. There are many generic types of errors that can apply to bcopy(). These include * mis-specifying the size of a buffer or the amount of data to be written. Off-by-one errors are common. * failing to plan for correct behavior when input is larger than expected. * assuming the wrong semantics for a parameter that controls data transfer and prevents buffer overflows. Because various functions use the buffer size, buffer size minus one, the remaining space in the buffer, etc., it is important to understand the bounding semantics for each function. In the context of strings: * Failing to allow space for a terminating null character. * Failing to ensure that a terminating null character is present; many standard functions consistently experience this failure.

Application Programming Interfaces

[1] daisy:35 (Barnum, Sean)

Function Name	Comments
bcopy()	This function is deprecated -- use memcpy in new programs.

Method of Attack

Bounds checking and off-by-one errors create opportunities for buffer overflow or denial of service attacks.

Solutions

Applicability	Description	Efficacy
Always	Always use #define or other const for declaration of size	Effective to the degree that consistent care is used.
Always	Always use SAME #define or const when checking bound sizes	Effective to the degree that consistent care is used.
Always	For strings, use (buffer size) - 1 to ensure space to put terminating \0	Effective to the degree that consistent care is used.
Always	For strings, always write a \0 to upper bound the buffer after processing a string	Effective to the degree that consistent care is used.
Always	Do a bounds check to verify that the buffer you are passing in is as big as you say and that it is big enough to hold the new contents. Verify that the returned buffer is null terminated.	Effective to the degree that consistent care is used.
Always	Identifying or writing safer versions of utility functions which incorporate checks, and then using these safer functions consistently improves safety.	Effective to the degree that consistent care is used.

Examples of Incorrect Code

- Example 1

```
int count[10] = {0, 1, 2, 3, 4, 5,  6, 7, 8, 9};
int ids[3]={0, 1, 2};
bcopy(count, ids, 5);
```

Examples of Corrected Code

- Example 1

```
const int SOURCE_BUFFER_SIZE = 10;
const int DESTINATION_BUFFER_SIZE = 3;
int count[SOURCE_BUFFER_SIZE] = {0, 1, 2, 3, 4, 5,  6, 7, 8, 9};
int ids[DESTINATION_BUFFER_SIZE] = {0, 1, 2};

/* limit number of integers to be copied */
bcopy(count, ids, DESTINATION_BUFFER_SIZE*sizeof(int));
```

Source References

- http://security-protocols.com/unixmanpages/bcopy.3.html
- ITS4 Source Code Vulnerability Scanning Tool - http://www.cigital.com/its4/

Discriminant Set

Operating Systems

- UNIX (All)

Languages

- C
- C++

Cigital, Inc. Copyright

####

###	########
Copyright Holder	Cigital, Inc.

####

[1] mailto:copyright@cigital.com

###	########
Attack Categories	Denial of Service Malicious Input
Operating System	UNIX (all)
Software Context	String Management
Vulnerability Categories	Buffer Overflow No Null Termination